Weighted Model Counting Without Parameter Variables

No Author Given

No Institute Given

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

2 Pseudo-Boolean Functions

Notation. For any propositional formula ϕ over a set of variables X and $p, q \in \mathbb{R}$, let $[\phi]^p_q \colon 2^X \to \mathbb{R}$ be a pseudo-Boolean function defined as

$$[\phi]^p_q(Y) \coloneqq \begin{cases} p & \text{if } Y \models \phi \\ q & \text{otherwise} \end{cases}$$

for any $Y \subseteq X$.

Definition 1 (Operations). *Let $f, g \colon 2^X \to \mathbb{R}$ be pseudo-Boolean functions, $x, y \in X$, $Y = \{y_i\}_{i=1}^n \subseteq X$, and $r \in \mathbb{R}$. Operations such as addition and multiplication are defined pointwise as*

$$(f + g)(Y) \coloneqq f(Y) + g(Y), \quad \textit{and} \quad (f \cdot g)(Y) \coloneqq f(Y) \cdot g(Y).$$

Note that this means that binary operations on pseudo-Boolean functions inherit properties such as associativity and commutativity. By not distinguishing between a real number and a pseudo-Boolean function that always returns that number, we can use the same definitions to define scalar *operations as*

$$(r + f)(Y) = r + f(Y), \quad \textit{and} \quad (r \cdot f)(Y) = r \cdot f(Y).$$

Restrictions $f|_{x=0}, f|_{x=1} \colon 2^X \to \mathbb{R}$ *of f are defined as*

$$f|_{x=0}(Y) \coloneqq f(Y \setminus \{x\}), \quad \textit{and} \quad f|_{x=1}(Y) \coloneqq f(Y \cup \{x\})$$

for all $Y \subseteq X$.

Projection $\exists_x$ *is an endomorphism $\exists_x \colon \mathbb{R}^{2^X} \to \mathbb{R}^{2^X}$ defined as*

$$\exists_x f \coloneqq f|_{x=1} + f|_{x=0}.$$

Since projection is commutative (i.e., $\exists_x \exists_y f = \exists_y \exists_x f$) [4, 5], we can define $\exists_Y \colon \mathbb{R}^{2^X} \to \mathbb{R}^{2^X}$ as $\exists_Y \coloneqq \exists_{y_1} \exists_{y_2} \dots \exists_{y_n}$. Throughout the paper, projection is assumed to have the lowest precedence (e.g., $\exists_x fg = \exists_x(fg)$).

Proposition 1 (Basic Properties). *For any propositional formulas ϕ and ψ, and $a, b, c, d \in \mathbb{R}$,*

- $[\phi]_b^a = [\neg\phi]_a^b$;
- $c + [\phi]_b^a = [\phi]_{b+c}^{a+c}$;
- $c \cdot [\phi]_b^a = [\phi]_{bc}^{ac}$;
- $[\phi]_b^a \cdot [\phi]_d^c = [\phi]_{bd}^{ac}$;
- $[\phi]_0^1 \cdot [\psi]_0^1 = [\phi \wedge \psi]_0^1$.

And for any pair of pseudo-Boolean functions $f, g \colon 2^X \to \mathbb{R}$ and $x \in X$, $(fg)|_{x=i} = f|_{x=i} \cdot g|_{x=i}$ for $i = 0, 1$.

References.

- Related work without publicly available implementations:
  - direct compilation to SDDs [3]
  - direct compilation to PSDDs, also eliminating parameter variables (a thesis)
  - maybe two more papers

TODO: Mention

- Weights on literals other than the positive literals that correspond to variables in X_P are redundant as they either are equal to one or duplicate an already-defined weight.
- Mention that formulas, clauses, and models are all treated as sets.

3 Weighted Model Counting

Definition 2 (WMC Instance [2] (TODO: not exactly though)). *A* WMC instance *is a tuple (ϕ, X_I, X_P, w), where X_I is the set of indicator variables, X_P is the set of parameter variables (with $X_I \cap X_P = \emptyset$), ϕ is a propositional formula in CNF over $X_I \cup X_P$, and $w \colon X_I \cup X_P \cup \{\neg x \mid x \in X_I \cup X_P\} \to \mathbb{R}$ is the weight function. The* answer *of the instance is $\sum_{Y \models \phi} \prod_{Y \models l} w(l)$.*

Remark 1. Encodings such as `cd05` and `cd06` are *not* WMC encodings. Instead, they encode Bayesian network inference into instances of the *minimum-cardinality* WMC problem, where the answer is defined to be $\sum_{Y \models \phi,\ |Y|=k} \prod_{Y \models l} w(l)$, where $k = \min_{Y \models \phi,\ Y \neq \emptyset} |Y|$, if k exists, otherwise the answer is zero. This additional condition on model cardinality becomes necessary because these encodings eliminate clauses of the form $p \Rightarrow i$, where $p \in X_P$ is a parameter variable, and $i \in X_I$ is an indicator variable. Nonetheless, our transformation algorithm still works on such encodings, although the experimental results are discouraging because they use approximately twice as many indicator variables. For instance, each binary variable of a Bayesian network is encoded using two indicator variables while one would suffice.

Definition 3 (PBP Instance). *A* pseudo-Boolean projection *(PBP) instance is a tuple (F, X, ω), where X is the set of variables, F is a set of pseudo-Boolean functions $2^X \to \mathbb{R}$, and $\omega \in \mathbb{R}$ is the scaling factor. The* answer *of the instance is $\omega \cdot \left(\exists_X \prod_{f \in F} f\right)(\emptyset)$.*

NOTE: The constant is inspired by the `bklm16` [1] encoding.

4 Parameter Variable Elimination

Notes.

- Let X_P be the set of parameter variable and X_I be the set of indicator variables.
- Parameter variables are either taken from the LMAP file (for encodings produced by Ace) or assumed to be the variables that have both weights equal to 1.
- If a parameter variable in a clause is 'negated', we can ignore the clause. We assume that there are no clauses with more than one instance of parameter variables.
- The second **foreach** loop can be performed in constant time by representing ϕ' as a list and assuming that the two 'clauses' are adjacent in that list (and incorporating it into the first loop).
- The d map is constructed in $\mathcal{O}(|X_P| \log |X_P|)$ time (we want to use a data structure based on binary search trees rather than hashing).
- `rename` can be implemented in $\mathcal{O}(\log |X_P|)$ time.
- This may look like preprocessing, but all the transformations are local and thus can be incorporated into an encoding algorithm with no slowdown. In fact, if anything, the resulting algorithm would be slightly faster, as it would have less data to output.

4.1 Proof of Correctness

Theorem 1 (Early Projection [4, 5], verbatim). *Let X and Y be sets of variables. For all functions $f\colon 2^X \to \mathbb{R}$ and $g\colon 2^Y \to \mathbb{R}$, if $x \in X \setminus Y$, then $\exists_x(f \cdot g) = (\exists_x f) \cdot g$.*

Lemma 1. *For any pseudo-Boolean function $f\colon 2^X \to \mathbb{R}$,*

$$(\exists_X f)(\emptyset) = \sum_{Y \subseteq X} f(Y).$$

Proof. For base case, let $X = \{x\}$. Then

$$(\exists_x f)(\emptyset) = (f|_{x=1} + f|_{x=0})(\emptyset) = f|_{x=1}(\emptyset) + f|_{x=0}(\emptyset) = f(\{x\}) + f(\emptyset) = \sum_{Y \subseteq \{x\}} f(Y).$$

This easily extends to $|X| > 1$ by the definition of projection on sets of variables.

Algorithm 1: WMC instance transformation

Data: WMC instance (ϕ, X_I, X_P, w)
Result: PBP instance (F, X, ω)

$F \leftarrow \emptyset$;
$\omega \leftarrow 1$;
let $d\colon X_P \to \mathbb{N}$ be defined as $p \mapsto |\{o \in X_P \mid o \leq p\}|$;
foreach *clause* $c \in \phi$ **do**
  if $c \cap X_P = \{p\}$ *for some* p **and** $w(p) \neq 1$ **then**
    if $|c| = 1$ **then**
      $\omega \leftarrow \omega \times w(p)$;
    else
      $F \leftarrow F \cup \left\{ \left[\bigwedge_{l \in c \setminus \{p\}} \neg l\right]_1^{w(p)} \right\}$;
  else if $\{p \mid \neg p \in c\} \cap X_P = \emptyset$ **then**
    $F \leftarrow F \cup \{[c]_0^1\}$;
foreach *indicator variable* $v \in X_I$ **do**
  if $\{[v]_1^p, [\neg v]_1^q\} \subseteq F$ *for some* p *and* q **then**
    $F \leftarrow F \setminus \{[v]_1^p, [\neg v]_1^q\} \cup \{[v]_q^p\}$;
replace every variable v in F with `rename(`v`)`;
return (F, X_I, ω);
Function `rename(`v`)`**:**
  $S \leftarrow \{u \in X_P \mid u \leq v\}$;
  if $S = \emptyset$ **then return** v;
  return $v - d(\max S)$;

Proposition 2. *Let* (ϕ, X_I, X_P, w) *be a WMC instance. Then*

$$\left(\left\{[c]_0^1 \mid c \in \phi\right\} \cup \left\{[x]_{w(\neg x)}^{w(x)} \;\middle|\; x \in X_I \cup X_P\right\}, X_I \cup X_P, 1\right) \tag{1}$$

is a PBP instance with the same answer.

Proof. The answer of Eq. (1) is

$$\begin{aligned}\left(\exists_{X_I \cup X_P} \left(\prod_{c \in \phi} [c]_0^1\right) \prod_{x \in X_I \cup X_P} [x]_{w(\neg x)}^{w(x)}\right)(\emptyset) &= \sum_{Y \subseteq X_I \cup X_P} \left(\left(\prod_{c \in \phi} [c]_0^1\right) \prod_{x \in X_I \cup X_P} [x]_{w(\neg x)}^{w(x)}\right)(Y) \\ &= \sum_{Y \subseteq X_I \cup X_P} \underbrace{\left(\prod_{c \in \phi} [c]_0^1\right)}_{f}(Y) \underbrace{\left(\prod_{x \in X_I \cup X_P} [x]_{w(\neg x)}^{w(x)}\right)}_{g}(Y)\end{aligned}$$

by Lemma 1. Note that

$$f(Y) = \begin{cases} 1 & \text{if } Y \models \phi, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad g(Y) = \prod_{Y \models l} w(l),$$

which means that

$$\sum_{Y \subseteq X_I \cup X_P} f(Y)g(Y) = \sum_{Y \models \phi} \prod_{Y \models l} w(l)$$

as required.

Theorem 2 (Correctness). *Algorithm 1, when given a WMC instance (ϕ, X_I, X_P, w), returns PBP instance with the same answer, provided the following conditions are satisfied:*

1. *for all indicator variables $i \in X_I$, $w(i) = w(\neg i) = 1$,*
2. *and either*
   (a) *for all parameter variables $p \in X_P$, there is a non-empty family of literals $(l_i)_{i=1}^n$ such that*
      i. *$w(\neg p) = 1$,*
      ii. *$l_i \in X_I$ or $\neg l_i \in X_I$ for all $i = 1, \dots, n$,*
      iii. *and $\{c \in \phi \mid p \in c \text{ or } \neg p \in c\} = \{p \vee \bigvee_{i=1}^n \neg l_i\} \cup \{l_i \vee \neg p \mid i = 1, \dots, n\}$;*
   (b) *or for all parameter variables $p \in X_P$,*
      i. *$w(p) + w(\neg p) = 1$,*
      ii. *for any clause $c \in \phi$, $|c \cap X_P| \le 1$,*
      iii. *there is no clause $c \in \phi$ such that $\neg p \in c$,*
      iv. *if $\{p\} \in \phi$, then there is no clause $c \in \phi$ such that $c \neq \{p\}$ and $p \in c$,*
      v. *and for any $c, d \in \phi$ such that $c \neq d$, $p \in c$ and $p \in d$, $\bigwedge_{l \in c \setminus \{p\}} \neg l \wedge \bigwedge_{l \in d \setminus \{p\}} \neg l$ is false.*

Proof. By Proposition 2,

$$\left(\left\{[c]_0^1 \mid c \in \phi\right\} \cup \left\{[x]_{w(\neg x)}^{w(x)} \,\middle|\, x \in X_I \cup X_P\right\}, X_I \cup X_P, 1\right) \tag{2}$$

is a PBP instance with the same answer as the given WMC instance. By Definition 3, its answer is

$$\left(\exists_{X_I \cup X_P} \left(\prod_{c \in \phi} [c]_0^1\right) \prod_{x \in X_I \cup X_P} [x]_{w(\neg x)}^{w(x)}\right)(\emptyset) \tag{3}$$

Since both Conditions 2a and 2b ensure that each clause in ϕ has at most one parameter variable, we can partition ϕ into $\phi_* \coloneqq \{c \in \phi \mid \texttt{Vars}(c) \cap X_P = \emptyset\}$ and $\phi_p \coloneqq \{c \in \phi \mid \texttt{Vars}(c) \cap X_P = \{p\}\}$ for all $p \in X_P$. We can then use Theorem 1 to reorder (3) into

$$\left(\exists_{X_I} \left(\prod_{x \in X_I} [x]_{w(\neg x)}^{w(x)}\right) \left(\prod_{c \in \phi_*} [c]_0^1\right) \prod_{p \in X_P} \exists_p [p]_{w(\neg p)}^{w(p)} \prod_{c \in \phi_p} [c]_0^1\right)(\emptyset).$$

Let us first consider how the unfinished WMC instance (F, X_I, ω) after the loop on Lines 4 to 11 differs from (2). Note that Algorithm 1 leaves each $c \in \phi_*$ unchanged, i.e., adds $[c]^1_0$ to F. We can then fix an arbitrary $p \in X_P$ and let F_p be the set of functions added to F as a replacement of ϕ_p. It is sufficient to show that

$$\omega \prod_{f \in F_p} f = \exists_p [p]^{w(p)}_{w(\neg p)} \prod_{c \in \phi_p} [c]^1_0. \tag{4}$$

Note that under Condition 2a,

$$\bigwedge_{c \in \phi_p} c \equiv p \Leftrightarrow \bigwedge_{i=1}^{n} l_i$$

for some family of indicator variable literals $(l_i)_{i=1}^n$. Thus,

$$\exists_p [p]^{w(p)}_{w(\neg p)} \prod_{c \in \phi_p} [c]^1_0 = \exists_p [p]^{w(p)}_1 \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]^1_0.$$

If $w(p) = 1$, then

$$\exists_p [p]^{w(p)}_1 \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]^1_0 = \exists_p \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]^1_0 = \left. \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]^1_0 \right|_{p=1} + \left. \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]^1_0 \right|_{p=0}. \tag{5}$$

Since for any input, $\bigwedge_{i=1}^n l_i$ is either true or false, exactly one of the two summands in Eq. (5) will be equal to one, and the other will be equal to zero, and so

$$\left. \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]^1_0 \right|_{p=1} + \left. \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]^1_0 \right|_{p=0} = 1,$$

where 1 is a pseudo-Boolean function that always returns one. On the other side of Eq. (4), since $F_p = \emptyset$, and ω is unchanged, we get $\omega \prod_{f \in F_p} f = 1$, and so Eq. (4) is satisfied under Condition 2a when $w(p) = 1$.

If $w(p) \neq 1$, then

$$F_p = \left\{ \left[\bigwedge_{i=1}^{n} l_i \right]^{w(p)}_1 \right\},$$

and $\omega = 1$, and so we want to show that

$$\left[\bigwedge_{i=1}^{n} l_i \right]^{w(p)}_1 = \exists_p [p]^{w(p)}_1 \left[p \Leftrightarrow \bigwedge_{i=1}^{n} l_i \right]^1_0,$$

and indeed

$$\exists_p [p]^{w(p)}_1 \left[p \Leftrightarrow \bigwedge_{i=1}^n l_i \right]^1_0 = \left([p]^{w(p)}_1 \left[p \Leftrightarrow \bigwedge_{i=1}^n l_i \right]^1_0 \right)\Bigg|_{p=1} + \left([p]^{w(p)}_1 \left[p \Leftrightarrow \bigwedge_{i=1}^n l_i \right]^1_0 \right)\Bigg|_{p=0}$$
$$= w(p) \cdot \left[\bigwedge_{i=1}^n l_i \right]^1_0 + \left[\bigwedge_{i=1}^n l_i \right]^0_1 = \left[\bigwedge_{i=1}^n l_i \right]^{w(p)}_1.$$

This finishes the proof of the correctness of the first 'foreach' loop under Condition 2a.

Now let us assume Condition 2b. We still want to prove Eq. (4). If $w(p) = 1$, then $F_p = \emptyset$, and $\omega = 1$, and so the left-hand side of Eq. (4) is equal to one. Then the right-hand side is

$$\exists_p [p]^1_0 \prod_{c \in \phi_p} [c]^1_0 = \exists_p \left[p \wedge \bigwedge_{c \in \phi_p} c \right]^1_0 = \exists_p [p]^1_0 = 0 + 1 = 1$$

since $p \in c$ for every clause $c \in \phi_p$.

If $w(p) \neq 1$, and $\{p\} \in \phi_p$, then, by Condition 2(b)iv, $\phi_p = \{\{p\}\}$, and Algorithm 1 produces $F_p = \emptyset$ and $\omega = w(p)$, and so

$$\exists_p [p]^{w(p)}_{w(\neg p)} [p]^1_0 = \exists_p [p]^{w(p)}_0 = w(p) = \omega \prod_{f \in F_p} f.$$

The only remaining case is when $w(p) \neq 1$ and $\{p\} \notin \phi_p$. Then $\omega = 1$, and

$$F_p = \left\{ \left[\bigwedge_{l \in c \setminus \{p\}} \neg l \right]^{w(p)}_1 \middle| \, c \in \phi_p \right\},$$

so need to show that

$$\prod_{c \in \phi_p} \left[\bigwedge_{l \in c \setminus \{p\}} \neg l \right]^{w(p)}_1 = \exists_p [p]^{w(p)}_{1-w(p)} \prod_{c \in \phi_p} [c]^1_0.$$

We can rearrange the right-hand-side as

$$\begin{aligned}\exists_p [p]^{w(p)}_{1-w(p)} \prod_{c \in \phi_p} [c]^1_0 &= \exists_p [p]^{w(p)}_{1-w(p)} \left[\bigwedge_{c \in \phi_p} c \right]^1_0 = \exists_p [p]^{w(p)}_{1-w(p)} \left[p \lor \bigwedge_{c \in \phi_p} c \setminus \{p\} \right]^1_0 \\ &= w(p) + (1 - w(p)) \left[\bigwedge_{c \in \phi_p} c \setminus \{p\} \right]^1_0 = w(p) + \left[\bigwedge_{c \in \phi_p} c \setminus \{p\} \right]^{1-w(p)}_0 \\ &= \left[\bigwedge_{c \in \phi_p} c \setminus \{p\} \right]^1_{w(p)} = \left[\neg \bigwedge_{c \in \phi_p} c \setminus \{p\} \right]^{w(p)}_1 = \left[\bigvee_{c \in \phi_p} \neg (c \setminus \{p\}) \right]^{w(p)}_1 \\ &= \left[\bigvee_{c \in \phi_p} \neg \bigvee_{l \in c \setminus \{p\}} l \right]^{w(p)}_1 = \left[\bigvee_{c \in \phi_p} \bigwedge_{l \in c \setminus \{p\}} \neg l \right]^{w(p)}_1 .\end{aligned}$$

By Condition 2(b)v, $\bigwedge_{l \in c \setminus \{p\}} \neg l$ can be true for at most one $c \in \phi_p$, and so

$$\left[\bigvee_{c \in \phi_p} \bigwedge_{l \in c \setminus \{p\}} \neg l \right]^{w(p)}_1 = \prod_{c \in \phi_p} \left[\bigwedge_{l \in c \setminus \{p\}} \neg l \right]^{w(p)}_1$$

which is exactly what we needed to show. This ends the proof that the first loop of Algorithm 1 preserves the answer under both Condition 2a and Condition 2b. Finally, the loop on Lines 12 to 14 of Algorithm 1 replaces $[v]^p_1 [\neg v]^q_1$ with $[v]^p_q$ (for some $v \in X_I$ and $p, q \in \mathbb{R}$), but, of course,

$$[v]^p_1 [\neg v]^q_1 = [v]^p_1 [v]^1_q = [v]^p_q,$$

i.e., the answer is unchanged.

5 Experimental Results

Notes.

– Apparently, the DPMC paper [5] already shows that taking the first offered decomposition tree is best.

6 Conclusions

Notes.

– This just says that p is equivalent to a conjunction.
– The first group of conditions applies to `d02`, while the second group applies to `bklm16`.

- For `cd05` and `cd06`, condition $2bi$ should be replaced with $w(\neg p) = 1$.
- Benefits of having this proof in the paper:
  - It puts all encodings on a common ground.
  - It illustrates the convenience of our notation for reasoning about (certain types of) pseudo-Boolean functions.
  - It's too big and too important to be left for the appendix.
- Processors:
  - Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz
  - Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
  - Intel(R) Xeon(R) CPU E7-4820 v2 @ 2.00GHz
- Table captions should be placed above the tables.
- Figures should preferably be in EPS format.
- the environments 'theorem', 'definition', 'lemma', 'proposition', 'corollary', 'remark', and 'example' are defined in the LLNCS documentclass as well.

TODO

- Do I need to formally consider extending a pseudo-Boolean function to a bigger domain?
- check if each condition is actually used. Maybe turn this into a paragraph that gives an overview of the proof.
- condition 1 is necessary in both cases because we're ignoring the weights of indicator variables (explicitly acknowledge this)
- maybe do more summations over domains in proofs
- add the `cd05`/`cd06` correctness theorem

References

1. Bart, A., Koriche, F., Lagniez, J., Marquis, P.: An improved CNF encoding scheme for probabilistic inference. In: Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (eds.) ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016). Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 613–621. IOS Press (2016). https://doi.org/10.3233/978-1-61499-672-9-613
2. Chavira, M., Darwiche, A.: On probabilistic inference by weighted model counting. Artif. Intell. **172**(6-7), 772–799 (2008). https://doi.org/10.1016/j.artint.2007.11.002
3. Choi, A., Kisa, D., Darwiche, A.: Compiling probabilistic graphical models using sentential decision diagrams. In: van der Gaag, L.C. (ed.) Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 12th European Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7958, pp. 121–132. Springer (2013). https://doi.org/10.1007/978-3-642-39091-3_11
4. Dudek, J.M., Phan, V., Vardi, M.Y.: ADDMC: weighted model counting with algebraic decision diagrams. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. pp. 1468–1476. AAAI Press (2020), https://aaai.org/ojs/index.php/AAAI/article/view/5505

5. Dudek, J.M., Phan, V.H.N., Vardi, M.Y.: DPMC: weighted model counting by dynamic programming on project-join trees. In: Simonis, H. (ed.) Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12333, pp. 211–230. Springer (2020). https://doi.org/10.1007/978-3-030-58475-7_13